



**CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)**

**CENTRE FOR RESEARCH
ON TRANSPORTATION**

**UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE**



**CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7**

**TÉLÉPHONE: (514) 343-7575
TÉLEX: 05-24146 BIBPOLYTEC
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121**

91-05408/

CRT-769

**Approaches to Stochastic 0-1 Linear
Programming with Simple Recourse**

by

**Yihua Li
Michel Gendreau**

Centre de recherche sur les transports — Publication #769

June 1991

Dépôt légal – 2ème trimestre 1991

RÉSUMÉ

De nombreux problèmes pratiques peuvent être formulés sous la forme de problèmes linéaires en nombres entiers binaires, par exemple, des problèmes de choix d'investissements, de localisation d'usines, d'horaires et de transport. À l'heure actuelle, plusieurs de ces problèmes sont posés dans une forme déterministe, alors qu'une formulation stochastique représenterait plus correctement les incertitudes qui existent relativement à certains des coefficients qui les définissent. On utilise souvent des estimés ponctuels de ces coefficients de manière à s'en tenir à une formulation déterministe, bien que cette approche ne soit pas toujours justifiée. En fait, dans plusieurs cas, cette approximation mène à des solutions sous-optimales.

Le but de cette recherche est de développer des algorithmes tant exacts qu'heuristiques pour résoudre des programmes linéaires stochastiques binaires avec recours simple et de produire des logiciels implantant ces algorithmes. Cet article présente un algorithme exact basé sur le principe d'énumération implicite et deux méthodes heuristiques efficaces pour la résolution de ces problèmes.

Mots-clés: programmation stochastique; programmation linéaire binaire; algorithmes; méthodes heuristiques.

ABSTRACT

The literature is replete with practical problems which can be formulated as 0-1 linear integer programs. These include such problems as capital budgeting, plant location, scheduling and transportation. Currently many of these mathematical programming problems are formulated in a deterministic manner even though they would more aptly be formulated as stochastic programs to incorporate the inherent uncertainty that exists in certain of the coefficients. A common approach is to use point estimates to obtain a deterministic formulation though this generally is not appropriate. In fact in many instances this approximation may result in suboptimal solutions.

The aim of this research is to develop both heuristic and exact algorithms to solve stochastic 0-1 linear integer programs with simple recourse (SLIPSR) and to produce applicable software packages for these algorithms. This paper will present an algorithm based on implicit enumeration concepts to solve stochastic 0-1 programming problem optimally and also two other algorithms which have been developed to solve such problem heuristically but efficiently.

Keywords : stochastic programming; 0-1 linear programming; algorithms; heuristics.

INTRODUCTION

Many practical problems can be formulated as 0-1 programming, such as capital budgeting, project selection, network design, plant location, transportation and scheduling problems. Such problems are usually formulated as deterministic models, for example, in a linear 0-1 programming model: $\max \{c'x \mid Ax \leq b, x=0,1\}$, where all parameters A , b and c are given and fixed, which is due to model conventional simplification. In general, such a situation is not always obtained. For instance, for a newspaper problem, the demand is random; for the production planning problem, available resources might also be uncertain. The advantage of stochastic linear program (SLP) is to allow some coefficients in the model to be stochastic processes rather than constants, thus reflecting the nature of random situations in real world problems.

With the advances of computer hardware and software, in the last ten years, number of approaches have been developed for dealing with Stochastic Linear Programs (SLP) or Linear Programs under Uncertainty. Integer SLP's, however, have not received the algorithmic attention that continuous problems have. Section 1 of this paper briefly reviews this literature base. Section 2 gives a formulation of zero-one Stochastic Linear Integer Programming with Simple Recourse (SLIPSR). Optimal algorithms designed to solve such problems are presented in Section 3. In section 4 we give two heuristic algorithms: HUSI and TABU respectively. Section 5 presents extensive computational comparison results between these two algorithms. Finally, a conclusion is given in Section 6.

Thus far we have developed some mathematical properties with respect to the solution set of SLIPSR (Li et al., 1989)^[9]. Using these results we initially intend to develop an algorithm to obtain an optimal solution to the SLIPSR problem using a

branch-and-bound approach.

This proposed algorithm will have two criteria for stopping rules. One to determine the condition where no further feasible solution exists along a branch and the other for the condition where no better solution exists along a given path.

Even though the feasible set for a SLIPSR is finite, the size of the problem that can be solved will be limited by using the exact algorithm. The complexity of the algorithm will be still $O(2^n)$. Heuristic approaches may therefore be a more practical solution method. Thus we have also been working on heuristic approaches to the SLIPSR problem.

In the first heuristic algorithm HUSI, we are developing a criterion to determine the selection of variables one by one and assign a value of 1 to them until a feasible solution has been reached and any further selection will increase the total cost. Then we develop a procedure to improve the objective function by dropping variables and introducing new ones.

In the second heuristic algorithm, we adopt the Tabu Search (TS) ideas proposed by Glover^{[5] [6]} to solve the problem. Generally speaking, this is a sort of improvement/exchange heuristic. It starts from an initial feasible solution, then changes from one feasible solution to another with least changing cost. For most local search methods, such as Clarke & Wright famous savings algorithm, the move might be stopped by a local optimum since no positive saving can be found at that point in the procedure. But in TS, even if a new solution is worse than the current one, such a change is still allowed. Obviously, those changes may cycle, so a list storing most recent changes is introduced to restrict any move which already exists in the list except if it brings better results. The procedure will be repeated until no further improvement can be reached within a maximum allowed number of iterations.

1. LITERATURE REVIEW

Optimization problems can be formulated as:

$$\begin{array}{ll}\text{Find } \mathbf{x} \text{ to} \\ \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \geq 0 \quad i=1, \dots, m\end{array} \quad (1)$$

where f and g_i are general functions of the decision variable vector $\mathbf{x} \in R^n$. If we know f and g_i exactly, we have deterministic (linear or nonlinear) optimization problems. However, when f and g_i are random or we do not have enough information about them, we have uncertain problems in which f and g_i are dependent on a pair of variables $(\mathbf{x}, \pi)$ where π is a random vector or an "environment-determining variable" [4]. The model of the problem can be written as:

$$\begin{array}{ll}\text{Find } \mathbf{x} \text{ to} \\ \text{minimize} & f(\mathbf{x}, \pi) \\ \text{subject to} & g_i(\mathbf{x}, \pi) \geq 0 \quad i=1, \dots, m\end{array} \quad (2)$$

where f and g_i are functions from Euclidean $n+p$ -space into the real line, $\mathbf{x}$ is an n -component vector of decision variables, and π is p -dimensional vector of random variables which may take on values in a set $\Omega \in R^p$. Now it is complicated to make a decision because one decision $\mathbf{x}$ may only be feasible for some, but not for all of $\pi \in \Omega$. The solution procedure is really dependent on the problem, i.e., on how much we know about the value of π . Hence, a number of solution methods have been

developed to deal with different kinds of stochastic optimization problems.

Two-stage stochastic linear programming was first introduced by Dantzig in 1961 for linear programs with random coefficients for decision making under uncertainty. Let X be the set of those x vectors which have the property that for every π in the set of its possible values, there exists at least one y (depending on x and π) such that $Tx + Iy = \pi$. The model of two-stage linear programming under uncertainty is formulated as

$$\begin{aligned}
 &\text{minimize} && c'x + E_{\pi}[\Phi(x,\pi)] \\
 &\text{subject to} && Ax \geq b \\
 &&& Tx + Iy = \pi \\
 &&& x \geq 0, \quad y \geq 0
 \end{aligned} \tag{3}$$

where c and x are n -component vectors, y and π are m_2 -component vectors, T is a $n \times m_2$ matrix, I is an identity matrix, b is an m_1 -vectors, and A is an $n \times m_1$ matrix. $\Phi(x,\pi)$ is the optimum value of the following linear programming problem (in which x and π are supposed to be fixed for a moment)

$$\begin{aligned}
 &\text{minimize} && p'y \\
 &\text{subject to} && Tx + Iy = \pi
 \end{aligned} \tag{4}$$

This is called the second stage problem in (3). The essential procedure of this algorithm is: the decision x is made, the values of the random variables occur, and then decisions y are made to compensate for any stage-one infeasibility. We solve the first stage problem only once but then the "experiment" producing the random variable π is "performed" many times. Thus we solve the second stage problem many

times and $E_{\pi}[\Phi(x, \pi)]$ is interpreted as the arithmetic mean of the obtained optima. If π has a discrete distribution, then we can obtain at once the first stage and the second stage optimal solutions.

Similar to deterministic problems, stochastic optimization problems can be classified as continuous or integer problems depending on whether problem consist of continuous variables or discrete variables. For instance, by specifying x in (3) to be integer we can have a stochastic integer programming model. However, for this model there is no efficient algorithm to solve applications of any realistic size of the problems^{[1] [9]}. An initial attempt to solve two-stage integer valued stochastic programs was proposed by Yudin and Tsoy^[11], but it turns out to be quite inefficient in solving 0-1 SLIPSR. Zimmermann and Pollatschek present a stochastic 0-1 program with randomly distributed objective function and right-hand-side coefficients^[12]. The methods they propose for solving the problem, however, may not turn out to be an efficient procedure in most cases according to the authors. Wollmer developed an algorithm to solve two stage linear integer programs under uncertainty where the first stage variables are integer and binary while the second stage variables are continuous^[10]. Unfortunately he did not report any computational experience.

The aim of this research is to develop both heuristic and exact algorithms to solve stochastic 0-1 SLIPSR and to produce applicable software packages for these algorithms. This paper will present an algorithm based on implicit enumeration concepts^[3] to solve stochastic 0-1 programming problem optimally and also other algorithms developed to solve such problem heuristically but efficiently.

2. FORMULATION

The stochastic 0-1 programming problem with simple recourse can be formulated as:

$$\text{Minimize: } z = \sum_{j=1}^n c_j x_j + \sum_{i=1}^m E[p_i^+ y_i^+ + p_i^- y_i^-] \quad (5)$$

(P)

$$\text{subject to: } \sum_{j=1}^n a_{ij} x_j \geq b_i, \quad i=1, \dots, m1 \quad (6)$$

$$\sum_{j=1}^n t_{ij} x_j + y_i^+ - y_i^- = \pi_i, \quad i=1, \dots, m2 \quad (7)$$

$$\text{all } x_j = 0 \text{ or } 1, \quad y_i^+, y_i^- \geq 0 \quad (8)$$

- where
- c_j — coefficient of x_j in objective function
 - a_{ij} — coefficient of x_j in the i^{th} constraint
 - b_i — the i^{th} deterministic resource
 - π_i — the i^{th} stochastic resource with a joint probability distribution $F(b)$
 - $\mathbf{x}$ — original decision vector
 - $\mathbf{y}^+$ — shortage vector
 - $\mathbf{y}^-$ — surplus vector
 - $\mathbf{p}^+$ — shortage penalty vector
 - $\mathbf{p}^-$ — surplus penalty vector

Suppose that π has distribution function $F(\pi)$ and let $F_i(\pi_i)$ be the marginal distribution of π_i . The expected penalty from the i th constraint is then given by

$$E[P(u, \pi)] = \sum_{i=1}^m p_i^+ [(\pi_i - t_i x)]^+ dF(\pi_i) + \sum_{i=1}^m p_i^- [(t_i x - \pi_i)]^+ dF(\pi_i) \quad (9)$$

where $[a]^+ = \max(a, 0)$, u is given by $Tx - u = 0$, or $Iy = \pi - u$, and $t_i = (t_{i1}, \dots, t_{in})$. It is clear that for each i the minimum expected penalty $P_i(u_i^*)$ in the second part right-hand-side of (9) can be obtained by setting u_i^* to be the solution of

$$\frac{d P(u_i^*)}{d u_i^*} = 0, \quad \text{if } P_i(u_i) \text{ is differentiable,}$$

or, if π_i is a discrete random variable with possible values $\{\pi_{ik}\}$, u_i^* can be set equal to one of the π_{ik} 's which can be determined through simple computations.

To see this, first note that it can never be optimal to take $u_i < \min\{\pi_{ik}\}$ or $u_i > \max\{\pi_{ik}\}$. Assume now that the π_{ik} 's are indexed in increasing order and that $\pi_{ik^*} < u_i^* < \pi_{i, k^*+1}$ for some k^* . The expected correction cost for constraint i is:

$$\begin{aligned} \text{correction cost} &= [p_i^- \sum_{k \leq k^*} p_{ik} (u_i^* - \pi_{ik}) + p_i^+ \sum_{k > k^*} p_{ik} (\pi_{ik} - u_i^*)] \\ &= p_i^- \sum_{k \leq k^*} p_{ik} [(u_i^* - \pi_{ik^*}) + (\pi_{ik^*} - \pi_{ik})] + p_i^+ \sum_{k > k^*} p_{ik} [(\pi_{ik} - \pi_{ik^*}) - (u_i^* - \pi_{ik^*})] \\ &= p_i^- \sum_{k \leq k^*} p_{ik} (\pi_{ik^*} - \pi_{ik}) + p_i^+ \sum_{k > k^*} p_{ik} (\pi_{ik} - \pi_{ik^*}) + (u_i^* - \pi_{ik^*}) [p_i^- \sum_{k \leq k^*} p_{ik} - p_i^+ \sum_{k > k^*} p_{ik}] \end{aligned}$$

$$\text{If } p_i^- \sum_{k \leq k^*} p_{ik} \geq p_i^+ \sum_{k > k^*} p_{ik},$$

then the expected correction cost is minimized by taking $u_i^* = \pi_{ik^*}$, while if the opposite is true, it is optimal to set $u_i^* = \pi_{i, k^*+1}$.

After we have determined u^* , we try to find the solution x^* to minimize the objective function $c'x^*$ and make Tx^* as close as possible to u^* to minimize the expected penalty cost. We now describe an algorithm for this problem.

3. EXACT ALGORITHM

This algorithm will find a vector $\mathbf{x}$ based on the implicit enumeration to optimize the objective function $c \mathbf{x}$ and the "correction cost" under the deterministic constraints $\sum a_{ij}x_j \geq b_i, \quad i=1, \dots, m1$. Note that in the following, we assume without loss of generality that $c \geq 0$ (if for $x_j, c_j < 0, x_j$ is replaced by the complementary variable $\bar{x}_j = 1 - x_j$).

The key of the algorithm here is to set two critical criteria which allow us to possibly stop the search before reaching the leaves of the search tree.

At the starting point, all x_j 's are set equal to zero. Then we have the two criteria:

1) No feasible solution criterion:

If by setting all remaining free variables to 1, the left-hand-side of any deterministic inequality constraint is still less than b_i , then no feasible solution exists on this branch.

2) No better solution criterion:

If the solution obtained at a node is feasible and for each stochastic constraint i , either $\sum_j t_{ij} x_j \geq u_i^*$ and $t_{ij} \geq 0$ for all nonbasic variables, or $\sum_j t_{ij} x_j \leq u_i^*$ and $t_{ij} \leq 0$ for all nonbasic variables, then there exists no better solution along this branch since any further search will let more variables enter into solution with value of 1 and it will increase the cost of objective function not only in its first part — $c \mathbf{x}$ but also the second part as it exceeds the expected optimal point u_i^* .

Denote

D_i , the current value of the left-hand-side for the i^{th} deterministic constraint,

$$D_i = \sum_{j=1}^n a_{ij}x_j - b_i, \quad i=1, \dots, m1; \quad (10)$$

d_{ij} , the reduction in the infeasibility measure obtained by setting x_j to 1,

for $j \in \{J \mid x_j=0\}$, $i=1, \dots, m1$,

$$d_{ij} = \begin{cases} \min(a_{ij}, -D_i) & \text{if } D_i < 0 \\ -\min(0, D_i + a_{ij}) & \text{otherwise;} \end{cases} \quad (11)$$

U_i , the deviation from u_i^* for the i^{th} stochastic constraint,

$$U_i = \sum_{j=1}^n t_{ij}x_j, \quad i=1, \dots, m2; \quad (12)$$

Tc_{ij} , the increase in the expected penalty cost for the i^{th} stochastic constraint if variable x_j is set equal to 1,

for $j \in \{J \mid x_j=0\}$, $i=1, \dots, m2$, if $t_{ij} > 0$,

$$Tc_{ij} = \begin{cases} -p_i^+ t_{ij}, & \text{if } U_i + t_{ij} \leq u_i^* \\ p_i^- (U_i + t_{ij} - u_i^*) - p_i^+ (u_i^* - U_i), & \text{if } U_i \leq u_i^* \leq U_i + t_{ij} \\ p_i^- t_{ij}, & \text{otherwise,} \end{cases} \quad (13)$$

if $t_{ij} = 0$, $Tc_{ij} = 0$,

if $t_{ij} < 0$, then

$$Tc_{ij} = \begin{cases} p_i^- t_{ij}, & \text{if } U_i + t_{ij} \geq u_i^* \\ p_i^+ (-U_i - t_{ij} + u_i^*) - p_i^- (-u_i^* + U_i), & \text{if } U_i + t_{ij} \leq u_i^* \leq U_i \\ -p_i^+ t_{ij}, & \text{otherwise,} \end{cases} \quad (13')$$

The criterion G_j is also established for determining the entering variable,

$$\begin{aligned}
 & \text{for } j \in \{J \mid x_j = 0\}, \\
 G_j = & \begin{cases} (c_j + \sum_{i=1}^{m2} Tc_{ij}) - M(\sum_{i=1}^{m1} d_{ij}) & \text{if any } D_i < 0 \\ M & \text{or } (c_j + \sum_{i=1}^{m2} Tc_{ij}) < 0 \\ & \text{otherwise,} \end{cases} \quad (14)
 \end{aligned}$$

where M is a large enough positive number to emphasize the consideration of any violation of the deterministic constraints, and $m1$ and $m2$ are the numbers of deterministic and stochastic constraints respectively. Let:

$$G_{j*} = \min \{ G_j \} \quad (15)$$

Assign 1 to x_{j*} which has the minimum value G_{j*} , repeat the procedure until:

- i) If there is no feasible solution or no solution which is better than the current existing one, stop and switch to another branch.
- ii) If all possible branches have been searched, i.e. $G_{j*} = M$, terminate.

Since the possible solution set is finite, the implicit enumeration procedure will terminate after a finite number of steps.

The detailed procedure for the algorithm is the following:

Step 1. Initialize

Let $M = 10.E36$ and set all $x_j = 0$.

Step 2. Calculate

$$z = \sum_{j=1}^n c_j x_j + \sum_{i=1}^{m2} E[p_i^+ y_i^+ + p_i^- y_i^-] \quad (5)$$

Denote $BEST$ as the best current objective function value. If $z \leq BEST$, set $BEST = z$; if all possible branches have been searched, go to step 7. Otherwise:

Step 3. Calculate:

$$D_i = \sum_{j=1}^n a_{ij}x_j - b_i, \quad i=1, \dots, m1; \quad (10)$$

for $j \in \{J | x_j=0\}$, $i=1, \dots, m1$,

$$d_{ij} = \begin{cases} \min(a_{ij}, -D_i) & \text{if } D_i < 0 \\ -\min(0, D_i + a_{ij}) & \text{otherwise;} \end{cases} \quad (11)$$

$$U_i = \sum_{j=1}^n t_{ij}x_j, \quad i=1, \dots, m2; \quad (12)$$

for $j \in \{J | x_j=0\}$, $i=1, \dots, m2$, if $t_{ij} > 0$,

$$Tc_{ij} = \begin{cases} -p_i^+ t_{ij}, & \text{if } U_i + t_{ij} \leq u_i^* \\ p_i^-(U_i + t_{ij} - u_i^*) - p_i^+(u_i^* - U_i), & \text{if } U_i \leq u_i^* \leq U_i + t_{ij} \\ p_i^- t_{ij} & \text{otherwise,} \end{cases} \quad (13)$$

if $t_{ij} = 0$, $Tc_{ij} = 0$,

if $t_{ij} < 0$, then

$$Tc_{ij} = \begin{cases} p_i^- t_{ij} & \text{if } U_i + t_{ij} \geq u_i^* \\ p_i^+(-U_i - t_{ij} + u_i^*) - p_i^-(-u_i^* + U_i), & \text{if } U_i + t_{ij} \leq u_i^* \leq U_i \\ -p_i^+ t_{ij} & \text{otherwise,} \end{cases} \quad (13')$$

for $j \in \{J | x_j=0\}$,

$$G_j = \begin{cases} (c_j + \sum_{i=1}^{m2} Tc_{ij}) - M(\sum_{i=1}^{m1} d_{ij}) & \text{if any } D_i < 0 \\ \text{or } (c_j + \sum_{i=1}^{m2} Tc_{ij}) < 0 & \\ M & \text{otherwise.} \end{cases} \quad (14)$$

Let $G_j = \min \{ G_j \}$. If $G_j = M$, go to step 5; otherwise:

Step 4. Assign value 1 to x_j and calculate:

$$D_i = D_i + a_{im}, \quad i = 1, \dots, m1; \quad U_i = U_i + t_{im}, \quad i=1, \dots, m2;$$

$$z = \sum_{j=1}^n c_j x_j + \sum_{i=1}^{m2} E[p_i^+ y_i^+ + p_i^- y_i^-] \quad (5)$$

Step 5. Feasible solution check

If any $D_i < 0$, $i = 1, \dots, m1$, then return to step 3 ;

Step 6. Better solution check

If $z \geq BEST$ and for each i , either $U_i \geq u_i^*$ and $t_{ij} \geq 0$ for all non-basic variables, or $U_i \leq u_i^*$ and $t_{ij} \leq 0$ for non-basic variables, reach back to the previous branching point and make a search on another subbranch. If all subbranches have been searched, go to step 7. Otherwise: go back to step 2.

Step 7. Terminate.

SAMPLE ILLUSTRATION

Consider the following problem :

$$\text{Min } z = 136x_1 + 60x_2 + 140x_3 + E[3y_1^+ + 2y_1^- + 3y_2^+ + 4y_2^-]$$

$$\text{s.t.} \quad 3x_1 + 2x_2 + 4x_3 \geq 6$$

$$2x_1 + 5x_2 + 3x_3 \geq 8$$

$$58x_1 + 67x_2 + 49x_3 + y_1^+ - y_1^- = \pi_1$$

$$76x_1 + 91x_2 + 80x_3 + y_2^+ - y_2^- = \pi_2$$

$$x_j = 0 \text{ or } 1, \quad y_i^+, y_i^- \geq 0$$

$\pi_1 = (100, 115, 130)$, shortage penalty $p_1^+ = 3$, surplus penalty $p_1^- = 2$,
 $\Pr \{\pi_{11} = 100\} = .3$, $\Pr \{\pi_{12} = 115\} = .5$, $\Pr \{\pi_{13} = 130\} = .2$.
 $\pi_2 = (130, 150)$, shortage penalty $p_2^+ = 3$, surplus penalty $p_2^- = 4$,
 $\Pr \{\pi_{21} = 130\} = .6$, $\Pr \{\pi_{22} = 150\} = .4$.

We explicitly calculate the value for u^* and we obtain:

$$u_1^* = 115,$$

$$u_2^* = 130.$$

Then we rewrite the problem as:

$$\text{Min } z = 136x_1 + 60x_2 + 140x_3 + E[3y_1^+ + 2y_1^- + 3y_2^+ + 4y_2^-]$$

$$\begin{aligned} \text{s.t.} \quad & 3x_1 + 2x_2 + 4x_3 \geq 6 \\ & 2x_1 + 5x_2 + 3x_3 \geq 8 \\ & 58x_1 + 67x_2 + 49x_3 + y_1^+ - y_1^- = 115 \\ & 76x_1 + 91x_2 + 80x_3 + y_2^+ - y_2^- = 130 \\ & x_j = 0 \text{ or } 1, \quad y_i^+, y_i^- \geq 0 \end{aligned}$$

Step 1. Set all $x_j = 0$, $M = 10.E36$ and $BEST = M$.

Step 2. Since the current solution ($x_1 = x_2 = x_3 = 0$) is infeasible, therefore

$$D_1 = -6, \quad D_2 = -8, \quad \text{and,}$$

$$U_1 = 0, \quad U_2 = 0.$$

Step 3. Calculate

$$d_{11} = 3, \quad d_{12} = 2, \quad d_{13} = 4$$

$$d_{21} = 2, \quad d_{22} = 5, \quad d_{23} = 3$$

$$Tc_{11} = -3 \times 58 = -174, \quad Tc_{12} = -3 \times 67 = -201, \quad Tc_{13} = -3 \times 49 = -147$$

$$Tc_{21} = -3 \times 76 = -228, \quad Tc_{22} = -3 \times 91 = -273, \quad Tc_{23} = -3 \times 80 = -240$$

$$G_1 = (136 - 174 - 228) - M(3 + 2) = -266 - 5M$$

$$G_2 = (60 - 201 - 273) - M(2 + 5) = -414 - 7M$$

$$G_3 = (140 - 147 - 240) - M(4 + 3) = -247 - 7M$$

$$G_j = \min \{ G_j \} = G_2 = -414 - 7M$$

Step 4. Assign value 1 to x_2 and calculate:

$$D_1 = -6 + 2 = -4, \quad D_2 = -8 + 5 = -3,$$

$$U_1 = 67, \quad U_2 = 91.$$

$$\begin{aligned} z &= 60 + [(100-67) \times 3 \times 0.3 + (115-67) \times 3 \times 0.5 + (130-67) \times 3 \times 0.2] \\ &\quad + [(130-91) \times 3 \times 0.6 + (150-91) \times 3 \times 0.4] \\ &= 60 + 139.5 + 131 = 330.5 \end{aligned}$$

Step 5. Feasible solution check. We should not stop here but go back to step 3 since the current solution is infeasible.

We continue the procedure until we get the optimal solution: $\mathbf{x}^* = (0,1,1)$,

$$\sum_{j=1}^3 a_{1j}x_j = 6$$

$$\sum_{j=1}^3 a_{2j}x_j = 8$$

$$y_1^- = 1 \quad \text{or} \quad \sum_{j=1}^3 t_{1j}x_j = 116$$

$$y_2^- = 21 \quad \text{or} \quad \sum_{j=1}^3 t_{2j}x_j = 171$$

The optimal objective function value is :

$$\begin{aligned} z &= 200 + [(116-100) \times 2 \times 0.3 + (116-115) \times 2 \times 0.5 + (130-116) \times 3 \times 0.2] \\ &\quad + [(171-130) \times 4 \times 0.6 + (171-150) \times 4 \times 0.4] \\ &= 200 + 19 + 132 = 351 \end{aligned}$$

4. HEURISTIC ALGORITHM

Introduction

This section has the practical goal of developing quick solution methods for zero-one SLIPSR. Its measure of success is empirically based. The following sections present two heuristic approaches to solve these problems, as well as a computational comparison of the efficiency of two algorithms. The computational time is bounded by a polynomial of the number of constraints and variables. A number of problems tested in this paper range from 50 to 1300 variables. The required solution time vary from 1 to 16129 seconds on a standard IBM compatible PC.

4.1 Problem Approach

Prior to formally presenting the algorithmic procedures of the two heuristic HUSI and TABU, we introduces the basic elements which underlie them. Essentially, these heuristics will be based on answers to the following three "where" questions^{[2][6]}. We use HUSI to illustrate this approach:

1) Where to start

As a starting point, we work on an initial solution (s) by setting all $x_j = 0$ ($j = 1, \dots, n$). Then based on criteria G_j (defined below), one of the x_j is selected to make a change from s . This process is then repeated until the stopping criteria is reached. Now, G_j is defined in such a fashion as to ensure the search is efficient and effective, i.e. the question "where to search" is answered.

2) Where to search

To select the appropriate variable to be changed in the solution vector, we choose the variable with the highest marginal contribution to the feasibility and the objective functions. Taking this observation into account, we can establish the criteria G_j for the algorithms in (17) and (19) respectively.

3) Where(when) to stop

If no further improvement can be reached, then terminate.

The motivations used to answer the three "where" questions now lead to a formal presentation of the heuristic algorithms.

4.2. HUSI Algorithm

We define the followings:

s — the current solution.

s_j — the new solution obtained by assigning a value 1 to the variable x_j and inserting x_j into s .

$z(s_j)$ — the new objective function value for s_j , which has been defined in (5).

The criteria G_j for variable selection is also established:

$$d_{1j} = \begin{cases} a_{1j} & \text{if } \sum_{k \neq j} a_{1k}x_k + a_{1j} < b_1 \\ b_1 - \sum_{k \neq j} a_{1k}x_k & \text{or if } \sum_{k \neq j} a_{1k}x_k < b_1 \leq \sum_{k \neq j} a_{1k}x_k + a_{1j} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$G_j = BEST - z(s_j) + M \left(\sum_{i=1}^{m1} d_{1j} \right), \quad j \in (J | x_j = 0) \quad (17)$$

Algorithm

Input: $n \times m1$ matrix A with a deterministic demand vector b ,
 $n \times m2$ matrix T with a stochastic demand vector π ,
Cost function which is defined in (5),
Probability distribution function P .

Output: A satisfactory solution.

Procedure:

(a) Initialization Step:

Set all $x_j = 0$, $j=1, \dots, n$, to define the initial solution s and let $BEST = z(s)$;

(b) Iterative Step:

Part 1: Calculate G_j in (17), $j=1, \dots, n$, let $G_{j^*} = \max \{ G_j \}$; where G_{j^*} is obtained by assigning a value 1 to the j^{th} variable and inserting into s to turn out a maximum contribution G_{j^*} .

Part 2: If $G_{j^*} < 0$, go to (c), otherwise:

Let: $x_{j^*} = 1$, $s = s_{j^*}$; if $z(s_{j^*}) < BEST$, let $BEST = z(s_{j^*})$;

Return back to (b);

(c) Improvement/Exchange Step:

Part 1: For each pair of $x_i \in X_1 = \{x \mid x=1\}$ and $x_j \in X_2 = \{x \mid x=0\}$, calculate the saving S_{ij} from the exchange $x_i \longleftrightarrow x_j$ (i.e. setting x_i to 0 and x_j to 1) and let $S = \max \{ S_{ij} \}$.

Part 2: If no positive saving exists (i.e. $S < 0$), stop, else go back to (c).

4.3. TABU Algorithm

In this section, s and $z(s_j)$ have the same definition as that in above section, but s_j is defined as a new solution obtained by reversing the variable x_j (i.e. setting $x_j = 1 - x_j$) in the current solution s , and the consequent reversing cost G_j is defined by

$$d'_{j} = \begin{cases} M & \text{if any } \sum_{k \in J} a_{1k}x_k - a_{1j}(1-x_j) < b_1 \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

$$G_j = z(s_j) + d'_{j} \quad j=1, \dots, n \quad (19)$$

The algorithm starts by assigning value 1 to each variable in the sequence of nondecreasing cost until an initial feasible solution reached. Then, for any least cost G_j obtained by complementing a variable x_j , if it is less than the current best objective function value or if the variable x_j is not in the Tabu list, stack the variable into the list, then update the solution by reversing the variable and the current best solution value. The procedure will be repeated until no improvement can be found within the maximum allowed iteration times. We define Q as a Tabu list of length of L , in which the variables which have been most recently reversed will be stored in a queue. This list will prevent the search from cycling. The Input and Output required in TABU are the same as one defined above.

Algorithm

1) Initialization Step:

Start from an initial feasible s solution obtained by assigning 1 to x_j in nondecreasing cost until the solution becomes feasible.

Select maximum allowable iterations(Max_iter), length of TABU list(L), then initialize TABU list and set $k = 0$, $BEST = z(s)$;

2) Iterative Step:

Part 1: Create a Candidate List with G_j obtained by reversing each variable:

$x_j = 1 - x_j$, $j = 1, \dots, n$, and update the iteration counter by setting $k = k + 1$;

Part 2: Find the best admissible (x_j is not in Q or G_j is better than $BEST$) candidate— s_{j*} .

Part 3: Adopt s_{j*} as the new solution by setting $x_{j*} = 1 - x_{j*}$ and update (stack x_j on) the Tabu List Q . If G_{j*} is better than $BEST$, record it as a current $BEST$ and reset k to 0.

3) Stopping Rule:

If $k > \text{Max\_iter}$, stop. Otherwise return to the iterative step.

An illustration of an application of the algorithm is detailed in the following section by way of the sample problem.

SAMPLE ILLUSTRATION

Still consider the sample problem of the previous section. The initial feasible solution s is: $x_j = 1$, $j = 1, \dots, 3$, $BEST = 893$. Then, x_j is reversed by setting it to 0, $BEST = 351$. The procedure will continue until the number of iterations specified (Max_iter) is reached. This result is the same as that from the optimal algorithm.

5. COMPUTER RESULTS

We now provide some computational characteristics concerning these two algorithms. The computational comparisons in the literature are often based on observed differences in average solution time, number of required iterations, etc. and usually based on well known published problem sets. In our work, the computational results are based on randomly generated problems. We believe that this method of comparison is better suited for making inferential statements. The table below provides a summary of results that compares the performance between the two proposed algorithms.

Table 1. Comparison of results between the two heuristic methods.

Number of Variables	Number of Iterations		Average CPU Time (sec.)		Average Cost		T/H (%) Average Cost
	HUSI	TABU	HUSI	TABU	HUSI	TABU	
50	32	31	13	3	706	616	87.3
100	60	59	45	19	1344	1062	79.0
150	95	94	104	57	1817	1375	75.7
200	123	122	183	79	2546	1922	75.5
250	140	139	271	87	3002	2347	78.2
300	243	196	1318	515	3845	2879	74.9
400	253	226	769	560	4715	3715	78.8
500	282	280	1081	874	5974	4563	83.1
1000	579	532	5962	4224	11670	8978	77.0
1300	822	724	13009	12224	16129	12220	75.8

The problems were generated as follows. The cost coefficients c_j are random numbers between 0 and 99; each a_{ij} is a random number between 0 and 99; while the b_i 's are random numbers between 100 and 32768 depending on the number of variables in the problem. For each different size, there are more than ten problems were tested.

6. CONCLUSION

Both exact and heuristic algorithms for solving zero-one SLIPSR problems have been developed. The exact algorithm is based on the concept of implicit enumeration method and can only be used to solve the problems of relatively small size.

The heuristic algorithms can solve the large-scale zero-one SLIPSR problems within reasonable computer time. The latter algorithms cannot be guaranteed to reach an optimal solution for a problem. However, we have tested the algorithms on many randomly generated problems with 3 - 30 variables, and found that the optimal solutions obtained by the enumeration method are either identical or differ very little from the approximate solutions obtained by TABU. The computer results also indicate that the TABU heuristic can produce cost savings of 12-24% when compared with HUSI. From our experience, we found that the length of TABU list is not critical in terms of accuracy and time.

REFERENCES:

- [1] Apolloni, B. and F. Pezzella, "Confidence intervals in the solution of stochastic integer linear programming problems," *Ann. Oper. Res.* **1** (1984) 67-68.
- [2] Crowder, J., E. L. Johnson and M. W. Padberg, "Solving large-scale zero-one linear programming problems," *Oper. Res.* **31** (1983) 803-834.
- [3] Deckro, R.E., and E. P. Winkofsky, "Solving zero-one multiple objective programs through implicit enumeration," *Eur. J. Oper. Res.* **12** (1983) 362-374.
- [4] Ermoliev, Y. and R. Wets, eds., Numerical Techniques in Stochastic Optimization, Springer-Verlag, Berlin, (1988).
- [5] Glover, F., "Tabu search, part I", *ORSA J. on Computing*, Vol. 1, No. 3 (1989) 190-206.
- [6] Glover, F., "Tabu search, part II", *ORSA J. on Computing*, Vol. 2, No. 1 (1990) 4-32.
- [7] Li, Y., "Heuristic algorithm for large-scale zero-one goal programming problems," *Theory and Practice of Systems Eng.* **3** (1985) 73-75.
- [8] Li, Y. , D. Kira and M. Kusy, "A heuristic approach to large-scale stochastic 0-1 linear programming with multiple objectives (SZOGP)", Presented at the ORSA/TIMS Joint National Meeting, New York, October 1989.
- [9] Rinnoy Kan, A. and L. Stougie, "Stochastic Integer Programming," (1988) in [4]
- [10] Wollmer, R. D. "Two Stage linear programming under uncertainty with 0-1 integer first stage variables," *Math. Programming* **19** (1980) 279-288.
- [11] Yudin, D. B. and E. V. Tsoy, "Integer valued stochastic programming," *Eng. Cybernetics* **12** (1973) 1-8.
- [12] Zimmermann, H. J. and M. A. Pollatschek, "The probability distribution function of the optimum of a 0-1 linear program with randomly distributed coefficients of the objective function and the right-hand-side," *Oper. Res.* **23** (1975) 137-149.